SCHOOL OF COMPUTING

pyMethods2Test:

A dataset of Python tests mapped to focal methods

Idriss Abdelmadjid

Robert Dyer

Problem: How to map Python tests to focal methods?

Test file (`test_math_ops.py`)

```python
import unittest
from math_ops import square, cube, MathUtils

class TestMathOps(unittest.TestCase):
    def test_square(self):
        self.assertEqual(square(2), 4)
        self.assertEqual(square(-3), 9)

    def test_fact(self):
        utils = MathUtils()

        self.assertEqual(utils.factorial(0), 1)
        self.assertEqual(utils.factorial(5), 120)
```

Focal file (`math_ops.py`)

```python
def square(n):
    return n * n
def cube(n):
    return n ** 3

class MathUtils:
    def factorial(self, n):
        if n == 0:
            return 1
        return n * self.factorial(n - 1)

    def multiply(self, a, b):
        return a * b
```

Problem: How to map Python tests to focal methods?

Test file (`test_math_ops.py`)

```python
import unittest
from math_ops import square, cube, MathUtils

class TestMathOps(unittest.TestCase):
    def test_square(self):
        self.assertEqual(square(2), 4)
        self.assertEqual(square(-3), 9)

    def test_fact(self):
        utils = MathUtils()

        self.assertEqual(utils.factorial(0), 1)
        self.assertEqual(utils.factorial(5), 120)
```

Focal file (`math_ops.py`)

```python
def square(n):
    return n * n
def cube(n):
    return n ** 3

class MathUtils:
    def factorial(self, n):
        if n == 0:
            return 1
        return n * self.factorial(n - 1)

    def multiply(self, a, b):
        return a * b
```

Problem: How to map Python tests to focal methods?

Test file (`test_math_ops.py`)

```python
import unittest
from math_ops import square, cube, MathUtils

class TestMathOps(unittest.TestCase):
    def test_square(self):
        self.assertEqual(square(2), 4)
        self.assertEqual(square(-3), 9)

    def test_fact(self):
        utils = MathUtils()

        self.assertEqual(utils.factorial(0), 1)
        self.assertEqual(utils.factorial(5), 120)
```

Focal file (`math_ops.py`)

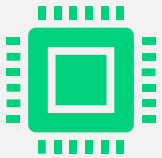
```python
def square(n):
    return n * n
def cube(n):
    return n ** 3

class MathUtils:
    def factorial(self, n):
        if n == 0:
            return 1
        return n * self.factorial(n - 1)

    def multiply(self, a, b):
        return a * b
```

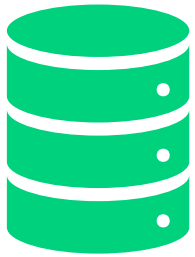
Applications

For Researchers:
Mining test smells, testing design patterns, test code styles, etc.



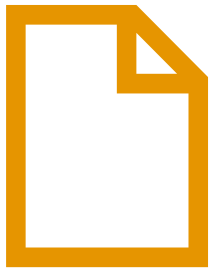
For Developers:
Could be used to train/fine-tune LLMs for automated test generation, fault localization, etc.

For Educators:
Real-world test examples for teaching and evaluation.



Key Statistics

Repositories:	88,846
Files:	
• Total:	18,517,737
• Test files:	1,289,630
Classes:	36,222,490
Methods:	
• Total:	222,020,293
• Test methods:	22,662,037
Mapped Focal Methods:	**2,198,378**



Data Format (JSON)

File data includes:

- **File paths**,
- **module name**,
- **methods**,
- **classes**,
- **line numbers**, and
- **indentation level**.

Test data also includes:

- **testing framework**,
- **non-library imports**, and
- **test methods.**

Focal data includes:

- **File paths**,
- **identified testing framework,**
- **test method locations**,
- **line numbers**,
- **indentation**, and
- **import statements**.

Solution

Test file (`test_math_ops.py`)

```python
import unittest
from math_ops import square, cube, MathUtils

class TestMathOps(unittest.TestCase):
    def test_square(self):
        self.assertEqual(square(2), 4)
        self.assertEqual(square(-3), 9)

    def test_fact(self):
        utils = MathUtils()

        self.assertEqual(utils.factorial(0), 1)
        self.assertEqual(utils.factorial(5), 120)
```

Focal file (`math_ops.py`)

```python
def square(n):
    return n * n
def cube(n):
    return n ** 3

class MathUtils:
    def factorial(self, n):
        if n == 0:
            return 1
        return n * self.factorial(n - 1)

    def multiply(self, a, b):
        return a * b
```

Solution

M. Tufano, S. K. Deng, N. Sundaresan, and A. Svyatkovskiy, "Methods2Test: A dataset of focal methods mapped to test cases," MSR 2022.

Test file (`test_math_ops.py`)

```python
import unittest
from math_ops import square, cube, MathUtils

class TestMathOps(unittest.TestCase):
    def test_square(self):
        self.assertEqual(square(2), 4)
        self.assertEqual(square(-3), 9)

    def test_fact(self):
        utils = MathUtils()

        self.assertEqual(utils.factorial(0), 1)
        self.assertEqual(utils.factorial(5), 120)
```

Focal file (`math_ops.py`)

```python
def square(n):
    return n * n
def cube(n):
    return n ** 3

class MathUtils:
    def factorial(self, n):
        if n == 0:
            return 1
        return n * self.factorial(n - 1)

    def multiply(self, a, b):
        return a * b
```

Solution

Test file (`test_math_ops.py`)

```python
import unittest
from math_ops import square, cube, MathUtils

class TestMathOps(unittest.TestCase):
    def test_square(self):
        self.assertEqual(square(2), 4)
        self.assertEqual(square(-3), 9)

    def test_fact(self):
        utils = MathUtils()

        self.assertEqual(utils.factorial(0), 1)
        self.assertEqual(utils.factorial(5), 120)
```

Focal file (`math_ops.py`)

```python
def square(n):
    return n * n
def cube(n):
    return n ** 3

class MathUtils:
    def factorial(self, n):
        if n == 0:
            return 1
        return n * self.factorial(n - 1)

    def multiply(self, a, b):
        return a * b
```

Solution

Test file (`test_math_ops.py`)

```
import unittest
from math_ops import square, cube, MathUtils

class TestMathOps(unittest.TestCase):
    def test_square(self):
        self.assertEqual(square(2), 4)
        self.assertEqual(square(-3), 9)

    def test_fact(self):
        utils = MathUtils()

        self.assertEqual(utils.factorial(0), 1)
        self.assertEqual(utils.factorial(5), 120)
```

Focal file (`math_ops.py`)

```
def square(n):
    return n * n
def cube(n):
    return n ** 3

class MathUtils:
    def factorial(self, n):
        if n == 0:
            return 1
        return n * self.factorial(n - 1)

    def multiply(self, a, b):
        return a * b
```

Solution

Test file (`test_math_ops.py`)

```python
import unittest
from math_ops import square, cube, MathUtils

class TestMathOps(unittest.TestCase):
    def test_square(self):
        self.assertEqual(square(2), 4)
        self.assertEqual(square(-3), 9)

    def test_fact(self):
        utils = MathUtils()

        self.assertEqual(utils.factorial(0), 1)
        self.assertEqual(utils.factorial(5), 120)
```

Focal file (`math_ops.py`)

```python
def square(n):
    return n * n
def cube(n):
    return n ** 3

class MathUtils:
    def factorial(self, n):
        if n == 0:
            return 1
        return n * self.factorial(n - 1)

    def multiply(self, a, b):
        return a * b
```

Sample test metadata (`math_ops.test.json`)


```
[
    {
        "file_path": "src/test_math_ops.py",
        "test_framework": "unittest",
        "test_imports": {},
        "test_methods": [
            {
                "method_name": "test_square",
                "line": 5,
                "line_end": 7,
                "indent": 4,
                "called_methods": [
                    "square"
                ]
            },
            {
                "method_name": "test_fact",
                "line": 9,
                "line_end": 12,
                "indent": 4,
                "called_methods": [
                    "utils.factorial"
                ]
            }
        ]
    }
]
```

Sample focal metadata (`math_ops.focal.json`)

```
{
    "test_math_ops.py": {
        "focal_file": "src/math_ops.py",
        "methods": {
            "test_square": {
                "line": 5,
                "line_end": 7,
                "indent": 4,
                "focal_class": null,
                "focal_method": {
                    "line": 1,
                    "line_end": 2,
                    "indent": 0,
                    "name": "square"
                }
            },
...
```

```
...
            "test_factorial": {
                "line": 9,
                "line_end": 13,
                "indent": 4,
                "focal_class": "MathUtils",
                "focal_method": {
                    "line": 7,
                    "line_end": 10,
                    "indent": 4,
                    "name": "factorial"
                }
            }
        }
    }
}
```

Sample focal metadata (`math_ops.focal.json`)

```
{
    "test_math_ops.py": {
        "focal_file": "src/math_ops.py",
        "methods": {
            "test_square": {
                "line": 5,
                "line_end": 7,
                "indent": 4,
                "focal_class": null,
                "focal_method": {
                    "line": 1,
                    "line_end": 2,
                    "indent": 0,
                    "name": "square"
                }
            },
...
```

```
...
            "test_factorial": {
                "line": 9,
                "line_end": 13,
                "indent": 4,
                "focal_class": "MathUtils",
                "focal_method": {
                    "line": 7,
                    "line_end": 10,
                    "indent": 4,
                    "name": "factorial"
                }
            }
        }
    }
}
```

Sample focal metadata (`math_ops.focal.json`)

```
{
    "test_math_ops.py": {
        "focal_file": "src/math_ops.py",
        "methods": {
            "test_square": {
                "line": 5,
                "line_end": 7,
                "indent": 4,
                "focal_class": null,
                "focal_method": {
                    "line": 1,
                    "line_end": 2,
                    "indent": 0,
                    "name": "square"
                }
            },
...
```

```
...
            "test_factorial": {
                "line": 9,
                "line_end": 13,
                "indent": 4,
                "focal_class": "MathUtils",
                "focal_method": {
                    "line": 7,
                    "line_end": 10,
                    "indent": 4,
                    "name": "factorial"
                }
            }
        }
    }
}
```

Sample focal metadata (`math_ops.focal.json`)

```
{
    "test_math_ops.py": {
        "focal_file": "src/math_ops.py",
        "methods": {
            "test_square": {
                "line": 5,
                "line_end": 7,
                "indent": 4,
                "focal_class": null,
                "focal_method": {
                    "line": 1,
                    "line_end": 2,
                    "indent": 0,
                    "name": "square"
                }
            },
...
```

```
...
            "test_factorial": {
                "line": 9,
                "line_end": 13,
                "indent": 4,
                "focal_class": "MathUtils",
                "focal_method": {
                    "line": 7,
                    "line_end": 10,
                    "indent": 4,
                    "name": "factorial"
                }
            }
        }
    }
}
```

Context for LLMs

```
def test_fact(self):
    utils = MathUtils()

    self.assertEqual(utils.factorial(0), 1)
    self.assertEqual(utils.factorial(5), 120)
```

Provide additional context to train LLMs:

- **global function signatures**,
- **class method signatures**,
- **class/instance attributes**, and
- **the focal method body**.

Context for LLMs

```python
def square(n): ...
def cube(n): ...

class MathUtils:
    def factorial(self, n):
        if n == 0:
            return 1
        return n * self.factorial(n - 1)

    def multiply(self, a, b): ...
```

```python
def test_fact(self):
    utils = MathUtils()

    self.assertEqual(utils.factorial(0), 1)
    self.assertEqual(utils.factorial(5), 120)
```

Provide additional context to train LLMs:

- **global function signatures**,
- **class method signatures**,
- **class/instance attributes**, and
- **the focal method body**.

Conclusion

- **First large-scale Python test-to-focal dataset**
  - Provides comprehensive mappings between test methods and focal methods for Python.
  - Dataset is publicly available, supporting further research, test generation, and tool development.

https://go.unl.edu/pymethods2test

- **Future Directions**
  - Support more testing frameworks, beyond pytest and unittest.
  - Refine heuristics to improve the handling of atypical naming schemes and edge cases.